

REMARKS

Claims 1-24 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Provisional Double Patenting:

Applicants note that the Examiner has provisionally rejected certain claims of copending application 09/653,229 under the judiciary created doctrine of obviousness-type double patenting as being unpatentable over claims of this application. However, since copending application 09/653,229 is in condition for allowance except for the provisional double patenting rejection, Applicants have requested, pursuant to M.P.E.P. 804.I.B, that the Examiner allow copending application 09/653,229 to issue and make the double patenting rejection in this instant application if the Examiner still believes such a rejection is appropriate.

Section 102 Rejections:

The Office Action rejected claims 1, 9, 16, 17 and 24 under 35 U.S.C. § 102(b) as being anticipated by Hill et al. (U.S. Patent 5,511,197) (hereinafter "Hill"). Applicants traverse this rejection for at least the following reasons.

Regarding claim 1, the Examiner states, "[t]he stub and proxy objects sent and received messages, acting as message gates as claimed." Applicants disagree with the examiner's interpretation of Hill. Hill discloses stub and proxy objects that implement remote procedure calls and communicate through the marshalling and unmarshalling of pointers, method names, and parameters into messages by inserting their actual values in the message (Hill, column 1, lines 25-45; column 7, lines 19-25; column 8 lines 20-23). Hill discloses storing copies of actual values in a message as well as converting from one data format to another (Hill, column 1, lines 40-47), but Applicants can find no teaching

in Hill disclosing a system wherein each message gate is configured for sending and receiving messages for one of the clients in a data representation language.

In response to Applicants' previous arguments, the Examiner argues that the "broad concept of communicating messages in a data representation language as claimed relates to nothing more than a standard method or format of the messages communicated" and that "Hill disclosed such a limitation, stating that stub channels utilized a certain protocol to communicate, thus disclosing communication of messages in a data representation language." (Final Office Action dated May 14, 2004, Response to arguments, paragraph 24). The Examiner is incorrect. Sending messages in a data representation language is not a standard method or format, but is distinctly different from sending messages using, for example, stubs and proxies, RMI or another code based communication format. Data representation languages are known in the art as languages that are used to represent or describe data in documents. The eXtensible Markup Language is one example of a data representation language. Data representation languages are not used in the prior art for programmatic interfaces such as the stub and proxy in Hill. Thus, Hill clearly does not anticipate sending and receiving messages in a data representation language.

In contrast, Hill describes a remote procedure call (RPC) mechanism. In the prior art, such mechanisms are specifically not based on data representation languages. A data representation language is a specific type of language having a specific purpose for describing data documents, as is well known to those of ordinary skill in the art. In the prior art, data representation languages were only used to describe data documents or content (such as HTML or XML documents that may be displayed or processed in Internet communications). In Hill, clients use a programmatic (code-based) messaging RPC interface to invoke services. The messages in Hill are clearly not data representation language messages.

In light of the above remarks, Applicants assert that the rejection of claim 1 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar remarks as discussed above in regard to claim 1 apply to claims 9 and 17.

Claims 1-4, 8-12, 15, 17-20 and 23 were rejected under 35 U.S.C. § 102(b) as being anticipated by Serlet et al. (U.S. Patent 5,481,721) (hereinafter "Serlet"). Applicants traverse this rejection for at least the following reasons.

Regarding claim 1, the Examiner states that Serlet teaches "A proxy object provided a way to send and receive messages on behalf of object, thus providing message gates as claimed." Applicants assert that Serlet discloses proxies that communicate through Mach messages (Serlet, column 9, lines 20-25). A Mach message contains "a header followed by zero or more data objects. For efficiency, large array arguments are passed out-of-line with copy-on-write semantics" (Serlet, column 9, lines 5-9). Serlet teaches that "[t]he argument encoding for standard C data types is explicit and strictly pass-by-value, and maps substantially directly onto a Mach message" (Serlet, column 12, lines 46-49). There is clearly no disclosure in Serlet of a system wherein each message gate is configured for sending and receiving messages for one of the clients in a data representation language.

In response to Applicants' previous arguments, the examiner submits "that a data representation language as claimed is broadly interpreted as a standard format or method for communicating data" and that "Serlet disclosed such a limitation, stating that messages sent between a sender and receiver must be encoded in a form understandable by both sides, providing a type of protocol for communicating data." (Final Office Action dated May 14, 2004, Response to arguments, paragraph 25). Again, the Examiner is incorrect. Sending messages in a data representation language requires more than just a standard method or format. Claim 1 does not state that messages can be in any standard format. Instead, claim 1 recites messages in a data representation language. A data representation language is a specific type of language used for describing documents, not messages for a programmatic interface as in Serlet. Data representation language

messages are distinctly different from Serlet's Mach messages or messages sent using other code-based communication formats. Serlet clearly does not *anticipate* sending and receiving messages in a data representation language.

Applicants assert that, in light of the above remarks, the rejection of claim 1 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar remarks as discussed above in regard to claim 1 apply to claims 9 and 17.

Claims 1-4, 8-12, 15, 17-20 and 23 were rejected under 35 U.S.C. § 102(e) as being anticipated by Marcos et al. (U.S. Patent 6,347,342) (hereinafter "Marcos"). Applicants traverse this rejection for at least the following reasons.

Regarding claim 1, the Examiner states that Marcos teaches "A connection between client objects and server objects was created through proxy and stub objects, which allowed for sending and receiving messages, thus providing message gates as claimed." Applicants submit that Marcos does not disclose a system wherein each message gate is configured for sending and receiving messages for one of the clients in a data representation language. Marcos teaches using a distributed object model or protocol to forward messages (Marcos, column 6, line 66 to column 7, line 3). Marcos discloses using code that will "encode and decode an operation and its parameters into a compacted message format" (Marcos, column 2, lines 12-14) which the mediating component translates or maps from one object model to another (Marcos, column 7, lines 16-24). Thus, Marcos does not disclose that each message gate is configured for sending and receiving messages for one of said clients in a data representation language.

In response to Applicants' previous arguments, the Examiner contends that "Marcos disclosed the use of a protocol for communicating messages between client objects and server objects, thus disclosing message gates configured to communicate messages in a data representation language." (Final Office Action dated May 14, 2004, Response to arguments, paragraph 26). Applicants respectfully disagree and argue that sending messages in a data representation language requires more than just a standard

method or format, and is distinctly different from sending messages using OLE/COM, CORBA, or another code based communication format as in Marcos. Thus, Marcos clearly does not anticipate sending and receiving messages in a data representation language.

Further, Applicants note that the Examiner's argument amounts to stating that sending messages in a data representation language is inherent in Marcos' teaching. Applicants respectfully remind the Examiner that for a prior art reference to be shown to inherently teach something, the Examiner must show evidence that makes it "clear that the missing descriptive matter is *necessarily* present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill." (emphasis added) (M.P.E.P. § 2131.01, paragraph 4). Applicants submit that messages in a data representation language are not inherent in the protocol for communicating messages between client and server objects using NEXT's DO, COM, OLE, or CORBA, as disclosed by Marcos. A similar argument applies to the other rejections above.

In light of the above remarks, Applicants assert that the rejection of claim 1 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar remarks as discussed above in regard to claim 1 apply to claims 9 and 17.

Regarding claim 2, the Examiner states, "Marcos disclosed proxy and stub objects to have the ability to verify object type." Applicants disagree with the Examiner's interpretation of Marcos. Marcos fails to teach verifying messages according to a data representation language message *schema*. Marcos discloses using messages comprising methods and method arguments (Marcos, column 6, lines 62-63; column 4, lines 23-25; column 15, lines 27-28; column 15, lines 64 – column 16, lines 4). Marcos teaches translating arguments from one object model to another (Marcos, column 16, lines 42-50). Applicants can find no reference in Marcos to a data representation language message schema. Hence, according to Marcos, the type of an argument from one object model is compared with an expected type in another object model, but Applicants submit

that Marcos fails to teach a system wherein each message gate is configured to verify messages according to a data representation language message schema.

In response to Applicants' previous arguments, the Examiner argues, "in its broadest sense, a data representation language message schema relates to nothing more than a set format for communicating data that messages must adhere to in order for a process to interpret them" and that "Marcus taught the use of such a format, stating that objects could communicate using specific data types, thus providing a message schema." The Examiner further submits that "verification of these data types was implicitly disclosed, as Marcos disclosed the possibility of translating data types into a type compatible with a receiving object, the need for such translation clearly determined only through a verification of the message data type according to a given schema acceptable by the receiving object." (Final Office Action dated May 14, 2004, Response to arguments, paragraph 27). Applicants submit that verifying messages according to a data representation language message schema is very different than simply teaching that objects may communicate using specific data types and also very different from translating data types as taught by Marcos. Specifically, a message containing data in a specific data type can be translated to a different data type without verifying that the message conforms to a particular schema, such as according to a data representation language schema. In other words, message content can be translated between data types based purely on a mapping of data types and not imply verifying the message according to a data representation language schema. This is in fact the type of translation Marcos teaches. For instance, Marcos states, "[w]hen a message is directed to the server object via the proxy object, mediating component 204 performs a mapping, or translation of arguments for use by the server object." (Marcos, column 7, lines 16-19).

Additionally, Marcos teaches, "when a connection request is received ... mediating component 204 determines whether a server object ... can service the message" and continues to state, "mediating component 204 queries information available in object model B to determine whether a service object exists in that system and created in that model that can process the message using the arguments supplied by

the client object.” (Marcos, column 7, lines 6- 12). Thus, Marcos actually teaches that, rather than verify a message according to a data representation language schema, a mediating component searches for a server object that can process the message as is. Applicants assert that these are two very different concepts.

In light of the above remarks, Applicants assert that the rejection of claim 2 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar remarks as discussed above in regard to claim 2 apply to claims 3-4, 10-12 and 18-20.

Claims 1, 6-9, 14-17 and 22-24 were rejected under 35 U.S.C. § 102(b) as being anticipated by Kingdon (U.S. Patent 5,349,642). Applicants traverse this rejection for at least the following reasons.

Regarding claim 1, the Examiner states, “Kingdon disclosed a method and system for client/server communication. A client and server make use of client and server stubs connected to a transport for communicating with each other, thus providing paired message gates as claimed.” Applicants submit that Kingdon does not teach a system wherein each message gate is configured for sending and receiving messages for one of the clients in a data representation language. Kingdon teaches using a very specific message packet format consisting of a length header, a request code representing the particular type of procedure being requested by the client, and data (Kingdon, column 5, lines 27-36, and Figure 3A). Kingdon’s message packet format is clearly very different from a data representation language. Applicants can find no reference to Kingdon teaching the use of a data representation language. Applicants submit that Kingdon does not disclose a system wherein each message gate is configured for sending and receiving messages for one of the clients in a data representation language.

In response to Applicants’ previous arguments, the Examiner argues, “a data representation language as claimed is broadly interpreted as a standard format or method for communicating data” and further argues that “Kingdon thus disclosed communicating

messages in a data representation language, as messages were communicated in packets created with a specific format.” (Final Office Action dated May 14, 2004, Response to arguments, paragraph 28). Applicants respectfully disagree and argue that sending messages in a data representation language is not just a standard method or format, but is distinctly different from sending messages using Kingdon’s unique message packet format. Kingdon clearly does not anticipate sending and receiving messages in a data representation language.

In light of the above remarks, Applicants assert that the rejection of claim 1 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar remarks as discussed above in regard to claim 1 apply to claims 9 and 17.

Section 103(a) Rejection:

The Office Action rejected claims 5, 13 and 21 under 35 U.S.C. § 103(a) as being unpatentable over Marcos in view of Bergman et al. (U.S. Patent 6,564,263) (hereinafter “Bergman”). Applicants traverse this rejection for at least the following reasons.

Applicants assert that the rejection of claims 5, 13 and 21 is unsupported by the cited art for at least the reasons given above in regard to their respective independent claims.

Furthermore, in regard to claim 5, the Examiner states, “While Marcos did not specifically mention the use of XML, it would have been obvious to one of ordinary skill in the art to consider the use of such a format, as Bergman disclosed XML to be both portable, independent of operating environment, and advantageous for linking different modalities of content.” Applicants disagree with the Examiner. Marcos fails to teach a system wherein each message gate is configured to verify messages according to a data representation language message schema, wherein the data representation message schema comprises an eXtensible Markup Language (XML) schema. Marcos teaches

using a mediating component that “identifies the expected method specification and arguments for the server” (Marcos, column 6, lines 62-63) and “performs any necessary message translation” (Marcos, column 4, lines 18-23). Also, Marcos uses a proxy object that “determines the expected method identification and the number and type of arguments for the server object” (Marcos, column 4, lines 23-25). Under Marcos, the mediating component “carries out dynamic translation at run-time” (Marcos, column 7, lines 22-23). Thus, it would make no sense in Marcos to verify messages according to a data representation language message schema.

Applicants submit that Marcos teaches the translation of method invocations and arguments, hence code, between two object models. Applicants can find no teaching in Marcos wherein each message gate is configured to verify messages according to a data representation language message schema, and wherein the data representation language schema comprises an eXtensible Markup Language (XML) schema.

Bergman discloses XML as “a tagged markup language for representing hierarchical, structured data” (Bergman, column 14, lines 12-14) and that “XML is useful for specifying and maintaining relationships between different modalities and versions, etc. of content” (Bergman, column 14, 20-22) (emphasis added). Thus, Bergman supports Applicants arguments above, in regard to the various § 102 rejections, that the prior art teaches data representation languages, such as XML, to be used for representing data content, not for programmatic messaging. Applicants assert that it would be counterintuitive to use XML to represent code such as the method invocations disclosed in Marcos. In the prior art, such as Bergman, XML is used to represent hierarchical, structured data (i.e. content), not to translate code, such as the translated method invocations in Marcos.

In response to Applicants’ arguments regarding claim 5, the Examiner argues that Bergman does disclose the use of XML for describing method specifications and function arguments and specifically states, “Bergman disclosed it [is] necessary to translate content data in some cases, such translation performed by a proxy” and cites column 5,

lines 44-63 of Bergman. The Examiner's interpretation of Bergman is clearly incorrect. Applicants submit that the cited reference discusses only the translation of *content* data from a "content server system" in order for the same *content* to be "displayed on different platforms" and that this translation is "necessary since the display and processing capabilities of the various client devices ... may differ widely." Thus, this passage does not mention anything using XML for describing method specification or function arguments, as the Examiner asserts. Actually, this passage does not discuss how such translation takes place, nor how a proxy would determine the display and processing capabilities of a client device.

The Examiner further argues, "a data representation encapsulating the necessary arguments and methods to perform the translation were described in a description scheme object" and cites column 6, lines 39-67. Applicants submit, however, that Bergman specifically states that the representation captures "all possible modalities (e.g. features, characteristics, semantics, metadata, etc) that may arise in different applications or events" (Bergman, column 6, lines 39-44). In fact, the cited passage repeatedly refers to the "data model", and "multimedia content." Applicants can find no reference in the cited passage to "a data representation encapsulating the necessary arguments and methods" as the Examiner contends.

While Bergman does disclose that an InfoPyramid defines methods and/or criteria for generating, manipulating, transcoding, and otherwise transforming the source multimedia content, such transformations of multimedia content has no relation to verifying messages according to a data representation language message schema. Bergman gives no suggestion that the multimedia content is verified according to any schema, let alone a data representation language message schema.

Thus, Applicants assert that any combination of Marcos and Bergman would still fail to teach wherein each message gate is configured to verify messages according to a data representation language message schema, and wherein the data representation language schema comprises an eXtensible Markup Language (XML) schema.

In light of the above remarks, Applicants assert that the rejection of claim 5 is not supported by the prior art and withdrawal of the rejection is respectfully requested. Similar remarks as discussed above in regard to claim 5 apply to claims 13 and 21.

CONCLUSION

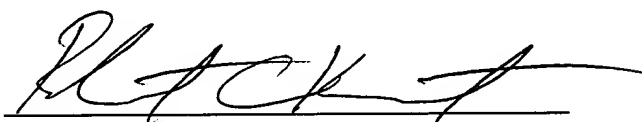
Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-64200/RCK.

Also enclosed herewith are the following items:

- ☒ Return Receipt Postcard
- ☐ Petition for Extension of Time
- ☐ Notice of Change of Address
- ☐ Fee Authorization Form authorizing a deposit account debit in the amount of \$
for fees ().
- ☐ Other:

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: July 13, 2004